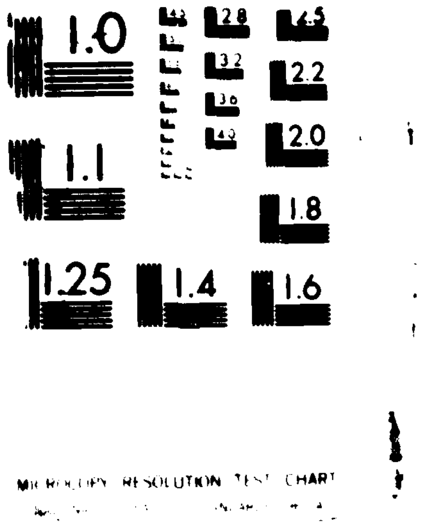
AD-A182 679 THE AUTOMATIC SYNTHESIS OF COMPUTER PROGRAMMING(U) 1/1
STANFORD UNIV CA DEPT OF COMPUTER SCIENCE Z MANNA
30 SEP 86 AFOSR-TR-87-0921 AFOSR-85-0383

UNCLASSIFIED F/G 12/5 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

AD-A182 679

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 87-0921	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE AUTOMATIC SYNTHESIS OF COMPUTER PROGRAMMING		5. TYPE OF REPORT & PERIOD COVERED Final scientific report 10/1/1985 - 9/30/1986
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Prof. Zohar Manna		8. CONTRACT OR GRANT NUMBER(s) AFOSR-85-0383
PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304 A3
CONTROLLING OFFICE NAME AND ADDRESS United States Air Force Air Force Office of Scientific Research Bldg. 410, Bolling Air Force Bases, Wash. DC 20332 NM		12. REPORT DATE / 13. NO. OF PAGES
		15. SECURITY CLASS. (of this report) unclassified
MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office) Same as 11		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
DISTRIBUTION STATEMENT (of this report) Approved for public release: distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from report) Capt. John Thomas		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Research on this effort was concentrated on the following topics: Special relations in automated deduction, binary-search algorithms, a theory of plans, deductive synthesis of dataflow networks, and temporal theorum proving. Two students received Ph.D.'s while conducting resesrch supported by this effort. Titles of several relevant papers produced during this grant include "Towards deductive synthesis of data-flow networks," "Non-clausal logic programming," "One origin of the binary-search paradigm," and "How to clear a block: a theory of plans."

87 7 28 374

AFOSR-TR- 87-0921

THE AUTOMATIC SYNTHESIS OF COMPUTER PROGRAMMING

by

Zohar Manna
Professor of Computer Science
Stanford University Stanford, CA 94305

Final Scientific Report:
Air Force Office of Scientific Research
Grant AFOSR-85-0383
Oct. 1, 1985 - Sept. 30, 1986

SUMMARY

Our research was concentrated on the following topics:

- Special Relations in Automated Deduction (Manna and Waldinger [86])

Theorem provers have exhibited super-human abilities in limited, obscure subject domains but seem least competent in areas in which human intuition is best developed. One reason for this is that an axiomatic formalization requires us to state explicitly facts that a person dealing in a familiar subject would consider too obvious to mention; the proof must take each of these facts into account explicitly. A person who is easily able to construct an argument informally may be too swamped in detail to understand, let alone produce, the corresponding formal proof. A continuing effort in our research is to make formal theorem proving more closely resemble intuitive reasoning. One case in point is our treatment of special relations.

In most proofs of interest for program synthesis, certain mathematical relations, such as equality and orderings, present special difficulties. These relations occur frequently in specifications and in derivation of proofs. If their properties are represented axiomatically, proofs become lengthy, difficult to understand, and even more difficult to produce or discover automatically. Axioms such as transitivity have many consequences, most of which are irrelevant to the proof; including them produces an explosion in the search space.

For the equality relation, the approach that was adopted early on is to represent its properties with rules of inference rather than axioms. In resolution systems, two rules of inference, paramodulation (Wos and Robinson) and E-resolution (Morris), were introduced. Proofs using these rules are shorter and clearer, because one application of a rule can replace the application of several axioms. More importantly, we may drop the equality axioms from the clause set, thus eliminating their numerous consequences from the search space.

We have discovered two rules of inference that play a role for an arbitrary relation analogous to that played by paramodulation and E-resolution for the equality relation. These rules apply to sentences employing a full set of logical connectives; they need not be in the clause form required

Codes

A-1

by traditional resolution theorem provers. We intend both these rules to be incorporated into theorem provers for program synthesis.

Employing the new special-relations rules yields the same benefits for an *arbitrary* relation as using paramodulation and E-resolution yields for equality: proofs become shorter and more comprehensible and the search space becomes sparser.

- Binary-Search Algorithms (Manna and Waldinger [85a])

Some of the most efficient numerical algorithms rely on a *binary-search* strategy; according to this strategy, the interval in which the desired output is sought is divided roughly in half at each iteration. This technique is so useful that some authors (e.g., Dershowitz and Manna, and Smith) have proposed that a general binary-search paradigm or schema be built into program synthesis systems and then specialized as required for particular applications.

It is certainly valuable to store such schemata if they are of general application and difficult to discover. This approach, however, leaves open the question of how schemata are discovered in the first place. We have found that the concept of binary search appears quite naturally and easily in the derivations of some numerical programs. The concept arises as the result of a single resolution step, between a goal and itself, using our deductive-synthesis techniques (Manna and Waldinger [80]).

The programs we have produced in this way (e.g., real-number quotient and square root, integer quotient and square root, and array searching) are quite simple and reasonably efficient, but are bizarre in appearance and different from what we would have constructed by informal means. For example, we have developed by our synthesis techniques the following real-number square-root program $sqrt(r, \mathcal{E})$:

$$sqrt(r,\ \epsilon) \ \Leftarrow \ \begin{cases} \mathit{if}\ max(r,\ 1) < \epsilon \\ \mathit{then}\ 0 \\ \mathit{else\ if}\ \left[sqrt(r,\ 2\epsilon)+\epsilon\right]^2 \le r \\ \qquad \mathit{then}\ sqrt(r,\ 2\epsilon)+\epsilon \\ \qquad \mathit{else}\ sqrt(r,\ 2\epsilon). \end{cases}$$

The program tests if the error tolerance ϵ is sufficiently large; if so, 0 is a close enough approximation. Otherwise, the program finds recursively an approximation within 2ϵ less than the exact square root of r. It then tries to refine this estimate, increasing it by ϵ if the exact square root is large enough and leaving it the same otherwise.

This program was surprising to us in that it doubles a number rather than halving it as the classical binary-search program does. Nevertheless, if the repeated occurrences of the recursive call $sqrt(r, 2\epsilon)$ are combined by common-subexpression elimination, this program is as efficient as the familiar one and somewhat simpler.

- A Theory of Plans (Manna and Waldinger [85b])

Problems in commonsense and robot planning were approached by methods adapted from our program-synthesis research; planning is regarded as an application of automated deduction. To support this approach, we introduced a variant of situational logic (Manna and Waldinger [81]), called *plan theory*, in which plans are explicit objects. A machine-oriented deductive-tableau inference system is adapted to plan theory. Equations and equivalences of the theory are built into a unification algorithm for the system. Frame axioms are built into the resolution rule.

Special attention was paid to the derivation of conditional and recursive plans. Inductive proofs of theorems for even the simplest planning problems, such as clearing a block, have been found to require challenging generalizations.

- Deductive Synthesis of Dataflow Networks (Jonsson, Manna, and Waldinger [86])

The synthesis of concurrent programs is much more complicated than the synthesis of sequential programs. In general, a concurrent program does not have a single input value and a single output value, but receives several inputs and sends several outputs during its execution. If we consider *sequences* of input and output values, then we can specify a concurrent program by giving a relation between the sequence of input values and the sequence of output values. This specification method is natural especially for networks of deterministic processes that communicate asynchronously by sending messages over buffered channels. Deterministic data flow networks fall into this category.

We have developed a method for the deductive synthesis of deterministic dataflow networks, which are specified by a relation between sequences of input values and sequences of output values.

Our synthesis method consists of two stages. The first stage, the deductive-synthesis stage, starts from a specification of the network. Using the deductive-tableau techniques of Manna and Waldinger [80], a system of recursive equations is synthesized. This system can be regarded as an applicative program that satisfies the specification for the network, but it does not directly represent any structure or parallelism of a network. In the second stage, the system of recursive equations is transformed into a dataflow network.

- The TABLOG Programming Language (Malachi, Manna, and Waldinger [85], Malachi [86])

We have developed a new logic-programming language, TABLOG (Malachi, Manna and Waldinger [84]). It is based on quantifier-free first-order logic that includes all the standard logical connectives, such as equality, negation, and equivalence. Programs are nonclausal: they do not need to be in Horn-clause form or any other normal form. They can compute either functions (as in LISP) or relations (as in PROLOG).

Two deduction rules are used for the execution of programs: *nonclausal resolution* (which corresponds to case analysis) and *equality replacement* (which corresponds to replacement of equals by equals). PROLOG programs are typically provided with cut annotations to allow their efficient execution. Such annotations are not necessary in TABLOG, since implicit cuts are introduced during the computation. Lazy evaluation provides an elegant way to manipulate infinite data structures.

A powerful mechanism has been introduced supporting a hierarchical structure for TABLOG programs and permitting the reuse of code. A compiler for a virtual TABLOG machine, written in TABLOG itself, is under development. It is expected that TABLOG programs will be executed as efficiently as their PROLOG counterparts, despite the additional features available to the programmer.

- Temporal Theorem Proving (Abadi and Manna [85], Abadi [86])

The concept of time occupies a central place in our understanding of computation. We often analyze computations as phenomena that occur over time, both formally and informally. Direct references to temporal notions can be avoided in some arguments about certain classes of systems, such as functional programs. However, processes that interact with an environment or with other processes are most naturally described in frameworks where time appears as an important explicit notion.

When classical logic serves as the framework to describe computations, time instants may be regarded as objects and represented as terms. An alternative is to put the concept of time at the core of a logic. The modal logics extend classical logic with modal operators to denote adverbs such as "necessarily" and "probably"; for a temporal theory of computation, the appropriate modal operators represent notions like "next," "always," and "eventually." Such a temporal logic may serve as an elegant and practical framework to reason about complex systems such as multiprocess programs.

In the last few years, temporal logic has been applied in the specification, verification, and synthesis of concurrent systems, as well as in the synthesis of robot plans and in the verification of hardware devices.

Many important properties of computation (e.g., termination, deadlock freedom, fairness) can be expressed directly and concisely in the language of temporal logic. This accounts for the convenience of temporal logic as a specification tool. Expressiveness does not always suffice, though. Some of the applications we mentioned involve a considerable deductive component. For example, the verification of a program typically includes proofs within temporal logic.

In our research, we have developed a novel proof system for temporal logic. The proof system is based on nonclausal resolution, a classical-logic method, and gives a special treatment to quantifiers and modal operators. We have explored soundness and completeness issues for this system and other related systems. In particular, we proved that a simple extension of the resolution system is as powerful as Peano Arithmetic. We also showed how to provide analogous resolution systems for other useful modal logics, such as the modal logics of knowledge and belief.

We have applied our resolution system to program verification. We have investigated the possibility that temporal logic would serve as a programming language and that a temporal-resolution theorem prover would interpret programs in this language.

- Logic: The Calculus of Computer Science

The research papers in which we have presented the deductive approach to program synthesis has been addressed to the usual academic readers of the scholarly journals. In an effort to make this work accessible to a wider audience, including computer science undergraduates and programmers, we have developed a more elementary treatment in the form of a two-volume book, *The Logical Basis for Computer Programming*, Addison-Wesley (Manna and Waldinger [85c]).

This book requires no computer programming and no mathematics other than an intuitive understanding of sets, relations, functions, and numbers; the level of exposition is elementary. Nevertheless, the text presents some novel research results, including

- theories of strings, trees, lists, finite sets and bags, which are particularly well suited to theorem-proving and program-synthesis applications;
- formalizations of parsing, infinite sequences, expressions, substitutions, and unification;
- a nonclausal version of skolemization;
- a treatment of mathematical induction in the deductive-tableau framework.

PUBLICATIONS

* Abadi, A. [86]

  Temporal Theorem Proving, Ph.D. thesis (supervised by Z. Manna), Computer Science Department, Stanford University, Stanford, CA, 1986.

* Abadi, A., and Z. Manna [85]

  Nonclausal temporal deduction, *Proceedings of the Logic of Programs Conference*, Brooklyn, NY, Lecture Notes in Computer Science 193, Springer-Verlag, June 1985, pp. 1–15.

* Jonsson, B., Z. Manna, and R. Waldinger [86]

  Towards deductive synthesis of data-flow networks, *First Symposium on Logic of Computer Science*, Cambridge, MA, June 1986, pp. 26–37.

* Malachi, Y. [86]

  Nonclausal Logic Programming, Ph.D. thesis (supervised by Z. Manna), Computer Science Department, Stanford University, Stanford, CA, 1986.

Malachi, Y., Z. Manna, and R. Waldinger [84]

  TABLOG: The deductive-tableau programming language, *ACM Symposium on* LISP *and Functional Programming*, Austin, TX, August 1984, pp. 323-330.

* Malachi, Y., Z. Manna, and R. Waldinger [85]

  TABLOG: Functional and relational programming in one framework, IEEE *Software*, Vol. 2, No. 1 (January 1986), pp. 75-76 (invited abstract).

Manna, Z., and R. Waldinger [80]

  A deductive approach to program synthesis, *ACM Transactions on Programming Languages and Systems*, Vol. 2, No. 1 (January 1980), pp. 90-121.

Manna, Z., and R. Waldinger [81]

  Problematic features of programming languages: A situational-calculus approach, *Acta Informatica*, Vol. 16, 1981, pp. 371-426.

* Manna, Z., and R. Waldinger [85a]

  The origin of the binary-search paradigm, *9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 222-224. Also to appear in *Science of Computer Programming.*

* Manna, Z., and R. Waldinger [85b]

  How to clear a block: A theory of plans, in *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*, Timberline, Oregon, July 1986, Morgan and Kaufmann. Also to appear in the *Journal of Automated Reasoning*, 1987.

* Manna, Z., and R. Waldinger [85c]

The Logical Basis for Computer Programming, Addison-Wesley, Reading, MA.

Volume 1: Deductive Reasoning (1985).

Volume 2: Deductive Techniques (to appear).

∗ Manna, Z., and R. Waldinger [86]

Special relat[illegible]n automated deduction, *Journal of the ACM*, Vol. 33, No. 1 (January 1986), pp. 1-6[illegible]

Papers marked by a ∗ were partially supported by the AFOSR grant. Copies of four of them are enclosed.